Construction of New Medicines via Game Proof Search

Abraham Heifets and **Igor Jurisica**
University of Toronto
Toronto, Ontario, Canada M5G 1L7
abe@cs.toronto.edu juris@ai.utoronto.ca

Abstract

The production of any new medicine requires solutions to many planning problems. The most fundamental of these is determining the sequence of chemical reactions necessary to physically create the drug. Surprisingly, these *organic syntheses* can be modeled as branching paths in a discrete, fully-observable state space, making the construction of new medicines an application of heuristic search. We describe a model of organic chemistry that is amenable to traditional AI techniques from game tree search, regression, and automatic assembly sequencing. We demonstrate the applicability of AND/OR graph search by developing the first chemistry solver to use proof-number search. Finally, we construct a benchmark suite of organic synthesis problems collected from undergraduate organic chemistry exams, and we analyze our solvers performance both on this suite and in recreating the synthetic plan for a multibillion dollar drug.

Introduction

Organic chemistry underlies medicine and materials science and provides modern conveniences in the forms of drugs, dyes, fragrances, and pesticides. A core task in organic chemistry is the planning of *organic syntheses*, the succession of chemical reactions that construct desired molecules from simple commercially-available starting materials. No medicine used in the developed world today would exist without solving these planning problems. Yet, despite 40 years of work by chemists on computer-aided organic synthesis (Corey and Wipke 1969; Law et al. 2009), the chemical synthesis planning problem is largely unknown in the AI community. In this paper, we bridge that gap by modeling organic synthesis planning as discrete state-space search and show that it is amenable to heuristic search techniques.

Specifically, we model the generation of organic syntheses as a two-player zero-sum game as follows: the first player picks a reaction that would synthesize the goal molecule, if the required reagents were available. Then, the second player picks one of that reaction's required reagents, and demands that the first player choose a reaction that can synthesize it. The players continue to take turns until the molecule to be synthesized is found in the library of starting materials, yielding a win for the first player. Alternatively, if no reactions can generate the chosen molecule, the second player wins. The game is a forced win for the first player if and only if she can construct all required reagents. Once all of the required reagents have been constructed, she can apply the original reaction and produce the goal molecule. This recursive construction game provides an algorithm to generate syntheses.

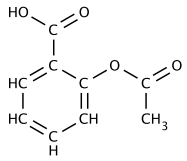

Figure 1: Aspirin. Vertices labeled C, H, or O denote carbon, hydrogen, and oxygen atoms, respectively. Single lines edges denote single bonds and double lines represent double bonds. Traditionally, bonds to hydrogen are not drawn.

Game proofs, which establish a game-theoretic value for game positions, closely match the structure we require in the production of organic synthesis. To demonstrate if a particular board configuration is a win for one player, a game tree proof provides, for every opponent move, a response that forces a win (although the responses can vary depending on the opponent's move). Game tree proofs can be computed even for games with complicated search spaces, such as games with multiple distinct paths to given board positions or reversible moves. The search space of organic chemistry forms a directed cyclical graph, with multiple synthetic paths between molecules and reactions pairs that are inverses of each other (*e.g.*, oxidations and reductions). In an organic synthesis, as in games of first-player loss (Kishimoto and Müller 2004), cycles denote that a molecule is consumed in its own production and such cyclic plans are not feasible. Proof number search and its memory-efficient variants (Allis, van der Meulen, and van den Herik 1994; Kishimoto 2010) have been used to generate game tree proofs for large real-world problems such as checkers (Schaeffer et al. 2007).

This paper provides three contributions. First, we introduce the AI community to organic synthesis planning, a long-standing problem of medical and economic importance. Historically, the effort of constructing chemistry-aware software carried too much overhead for AI researchers; today, however, a number of toolkits that encapsu-

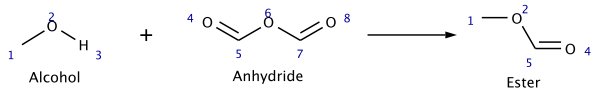

Figure 2: Esterification reaction of an alcohol active site with an anhydride active site to produce an ester. Atoms in the molecular fragments are numbered to help the reader track bond changes. When atomic labels are omitted, the vertex is presumed to be a carbon atom with sufficient hydrogens to total 4 bonds. Reaction conditions have been omitted for simplicity.

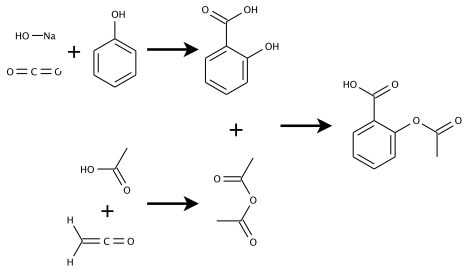

Figure 3: Synthesis of aspirin (right column) from carbon dioxide, sodium hydroxide, phenol, acetic acid and ketene starting materials (left column) via the precursor molecules, salicylic acid (top mid) and acetic anhydride (bottom mid). The final aspirin-forming step is an application of the esterification reaction depicted in Fig 2.

late chemical knowledge can be used as blackboxes. These include free open source software packages such as OpenBabel (O'Boyle et al. 2011) and RDKit (Landrum 2006). Second, we describe the first application of proof number search to chemical synthesis planning. Third, we offer the first public synthetic planning benchmark. To encourage further development in this essential area, we freely distribute both our solver and benchmark. We conclude with a discussion of the need for specific new algorithmic developments to address the challenges of the organic synthesis planning problem.

An Overview of Organic Synthesis

Organic synthesis presents an interesting and challenging planning domain due to the complexity of the state description, the cost of checking whether a state satisfies the goal conditions, and the number of operators and their interaction. Here we provide the AI practitioner with a qualitative description of these factors. Due to lack of space, we do not present all of organic chemistry here. For a more comprehensive survey of computer-aided organic synthesis in particular and organic chemistry in general, we refer the reader to (Todd 2005) and (Clayden et al. 2000), respectively.

Figures 1, 2, and 3 show typical depictions of a molecule, a reaction, and a multistep synthesis. The "game board state" of the organic synthesis game is a molecule. Molecules are represented by graphs where vertices correspond to atoms (of elements such as carbon, oxygen, hydrogen, etc.) and edges correspond to bonds of different types (single, double, aromatic, etc.). The "game moves" in chemistry are reactions, which describe the change in bonds that can be enacted on molecules. Reaction specifications include required activating substructures, which must be present to enable a reaction, and interfering substructures, which forbid them. These required and forbidden substructures are similar to the positive and negative preconditions of planning operators. Despite eliding many factors of chemical reactions, such as temperature, solvents, catalysts, and side products, such transformation-based reaction descriptions capture the structural modifications that are achievable through a reaction, and are therefore widely-used by chemists (Pirok et al. 2006; Daylight 2008).

It is important to note that the same reaction can be applied to many different molecules, provided that the molecules contain the appropriate reactive substructures. For example, the final step of Figure 3 is an application of the reaction described in Figure 2, and the alcohol is one small piece of salicylic acid. This applicability of reactions to many different molecules is similar to ungrounded operators from domain-independent planning.

Furthermore, one reaction may produce multiple possible products. Unlike deterministic games, where the same move applied to the same state produces a single outcome, a reaction can match several alternative reactive sites in a molecule. Consider again the reaction described in Figure 2. If it is applied to a molecule containing two alcohol groups, we could imagine 3 distinct product molecules, where the reaction has occurred at one alcohol, the other, or both.

Finally, one molecule may be produced from many precursors, even when using the same reaction. Every atom in the product originates in a reagent but the converse is not true because these reaction definitions are not assumed to be balanced or symmetric. While atoms are neither created nor destroyed in a chemical reaction, chemists typically isolate a single product of interest, effectively 'losing' atoms in side-products. In Figure 2, the atoms labelled 6, 7, and 8 are lost into solution and are not incorporated into the final ester product. Given this reaction definition, there exist an infinite number of molecules that could produce the ester: any molecule that contains an anhydride substructure can hang alternative substituents off of the 7 atom, and would not be constrained by the specification of the product and the reaction.

A particular challenge for computational chemistry is that molecules can be considerably larger than the states generally considered in computer game playing or automatic reasoning. Chromosome 1 in a typical human cell is a pair of molecules, each containing approximately 5.5 billion atoms (S. G. Gregory et al. 2006). Palytoxin, among the largest nonpolymeric molecules to be artificially synthesized, has 409 atoms (Hudlicky and Reed 2007). Any combination of constituent elements, graph connectivity, and 3-dimensional molecular configuration can define distinct molecules so it is unsurprising that many different molecules have been characterized in practice. Over 30 million molecules have been deposited in the PubChem compound database, of which 14 million come from chemical vendors (Wang et al. 2009). To determine whether a synthesis can begin from a given molecule because it is commercially available, an organic synthesis planner must efficiently check the molecule for membership in this set.

Chemical syntheses tend to be relatively short. Because a single reaction's yield is never 100%, long syntheses pro-

duce very little product. The average industrial pharmaceutical synthesis uses 8.1 steps, albeit with a heavy tail of syntheses with 16 or more reactions (Carey et al. 2006). Complex academic targets or natural products may require more involved chemistry. For example, synthesis of vitamin B_{12} required over 100 steps (Nicolaou and Sorensen 1996).

Additionally, the branching factor in the search for a synthesis can be large. ChemAxon provides a free-for-academic-use reaction engine containing a library of 145 manually-curated commonly-used reactions (ChemAxon 2011). A statistical analysis of synthesis databases estimated the existence of 92,781 unique known transformations (Law et al. 2009). Since the reaction patterns can match multiple distinct parts of the goal molecule, many distinct sets of precursor reagents can be proposed for each reaction (Agarwal, Larsen, and Gelernter 1978). However, the creation of reaction libraries is beyond the scope of this article. Our solver takes them as an input specified by domain experts.

A Proof Number Search-based Solver

Given a two-player zero-sum game-based formulation of chemical synthesis planning, it is natural to apply AND/OR graph search. AND/OR graph search has been applied to the derivation of winning strategies in two-player games (Schaeffer et al. 2007; Allis, van der Meulen, and van den Herik 1994; Nagai 1998; 2002) as well as problems such as MDPs (Bonet and Geffner 2006), assembly plans (Homem de Mello and Sanderson 1990), and web service composition (Liang and Su 2005). AND/OR graphs comprise two types of nodes that may be labeled as solved. An AND node is solved if all of its children are solved, while an OR node is solved if any of its children are solved. A solution graph therefore specifies (at least) one action for each non-goal node, which connect the initial state to some set of goal states.

For organic synthesis, an OR node corresponds to a molecule. The edges from the OR node correspond to the reactions that could produce the molecule. Each edge points to an AND node denoting the set of required reagents. The AND node has edges to the OR nodes that represent each required reagent molecule. In other words, the OR node representing a molecule that could be produced by 5 different reactions would have 5 edges pointing to the required reagents for each reaction. The molecule can be produced (*i.e.*, solved) if any one of the reactions is executed but each reaction is executable only if all of its required precursors are available. Molecules in the set of starting materials are labeled as solved, requiring that molecules not available as starting materials are considered solved if and only if they are produced by some reaction with solved reagents.

Proof number search, or PNS, analyzes two-player zero-sum games and determines whether a player has a forced win (Allis, van der Meulen, and van den Herik 1994). As a side-effect, it returns the strategy that produces the victory. In the context of PNS, a node is $proven$ if it is a guaranteed win for player 1 and $disproven$ if player 1 cannot prevent a loss. PNS maintains a proof number for each node, which correspond to the number of leaves in the subtree rooted at the node that would need to be examined to conclude that the node is proven. Similarly, each node has a disproof number that tracks the smallest number of leaves needed to be disproved for the node to be disproved.

Proof numbers are defined as follows:

$$\text{proof}(n) = \tag{1}$$
$$\begin{cases} 0 & \text{if } n \text{ is terminal win} \\ \infty & \text{if } n \text{ is terminal loss} \\ 1 & \text{if } n \text{ is unevaluated leaf} \\ \sum_{x \in \text{children}(n)} \text{proof}(x) & \text{if } n \text{ is internal AND node} \\ \min_{x \in \text{children}(n)} \text{proof}(x) & \text{if } n \text{ is internal OR node} \end{cases}$$

Disproof numbers are defined analogously:

$$\text{disproof}(n) = \tag{2}$$
$$\begin{cases} \infty & \text{if } n \text{ is terminal win} \\ 0 & \text{if } n \text{ is terminal loss} \\ 1 & \text{if } n \text{ is unevaluated leaf} \\ \min_{x \in \text{children}(n)} \text{disproof}(x) & \text{if } n \text{ is internal AND node} \\ \sum_{x \in \text{children}(n)} \text{disproof}(x) & \text{if } n \text{ is internal OR node} \end{cases}$$

No matter how many additional nodes are evaluated, a won node cannot be disproved; an AND node is disproven if any of its children are disproven; and an OR node is disproven only if all of its children are disproven.

Proof number search begins with an unevaluated start leaf node. The algorithm selects a leaf node, expands it, and assigns it proof and disproof numbers in accordance with Definitions 1 and 2. The selected leaf is found by descending from the root, at each point choosing the child node that needs the least work to (dis)prove it, until the leaf is found. The choice at an interior OR node is not governed by its children's disproof numbers; since every child needs to be disproved, the order in which they are examined is irrelevant. However, it is possible that a proof will be discovered for a child during exploration, which will prove the OR node and terminate exploration of the rest of the OR-node-rooted subtree. Therefore, the child selected at interior OR nodes is the one with the minimal proof number. By construction, this child will have a proof number equal to the OR node. The case for interior AND nodes is symmetric.

Once the node is expanded, its (dis)proof numbers are likely to have changed, so the proof numbers of this node and its ancestors must be updated. PNS walks node parent pointers, recalculating the proof and disproof numbers on the way up. This process is repeated until the root node is either proven (that is, its proof number equals 0) or disproven (disproof number = 0).

For organic synthesis, a root OR node is constructed for the goal molecule. Every reaction in the reaction library is checked to determine whether it could have produced the goal molecule. For each reaction that matches the goal, we generate possibly several sets of precursor molecules that could react to produce the goal. Each molecule in the set of precursors is mapped via a transposition table to an OR node, and then the set of precursor molecules is wrapped in an AND node. This AND node is connected as a child to

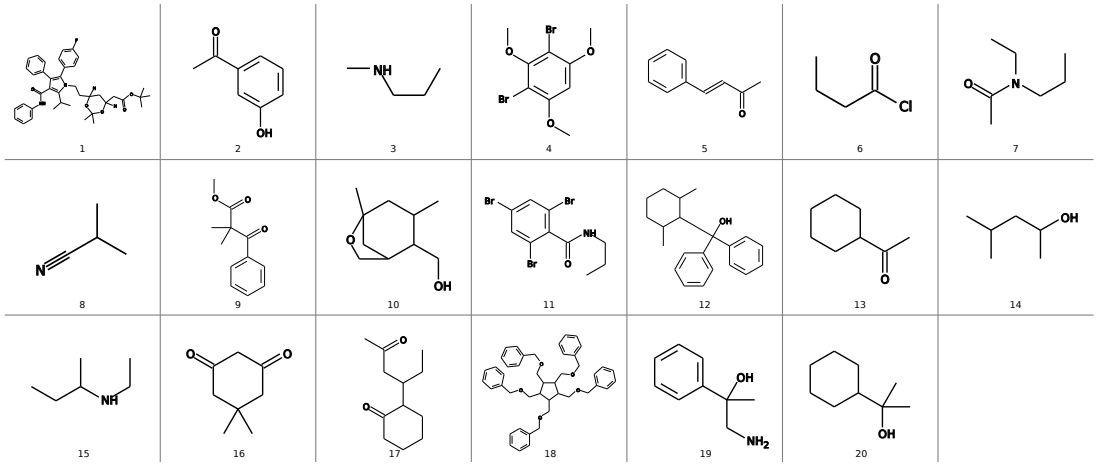

Figure 4: The benchmark target molecules. Images generated directly from the problem definition using OpenBabel (O'Boyle et al. 2011).

the original goal molecule's OR node. In the case where a reaction only requires a single precursor, we perform a simple memory optimization and use the precursor's OR node as a child directly, omitting the usual wrapping AND node. The proof and disproof numbers of the root node are updated according to Definitions 1 and 2.

Selection and expansion of leaves occurs as in normal PNS with several modifications. A leaf node is considered to be a terminal win if its molecule is found in the starting material library and it is considered to be a terminal loss if either no reactions would produce the molecule or if any required precursor set contains an ancestor molecule in the current synthesis path. ChemAxon's JChem 5.7.0 toolkit is used for all reaction applications and starting material matching (ChemAxon 2011). Where possible, the precursor molecules are not grounded but are interpreted as minimal requirements over matching sets of molecules. Although the subgraph isomorphism necessary for reaction matching is intractable in the worst case, domain-specific algorithms can be fast in practice (Cao, Jiang, and Girke 2008).

Usually, in these game proof algorithms, playing a move is considered to be free. This can lead to very large proof trees. Using the move-cost modification from dfpn$^+$ (Nagai 2002), we define the cost of each reaction application to be 1, to help focus the search on shallow convergent syntheses.

Definitions 1 and 2 imply that parent AND/OR nodes contain a child with equal (dis)proof counts. However, this condition only holds if the search space is a tree. Since the chemical search space is a graph, the search algorithm will modify the values of one node and update the values of only one of its parents. When the search encounters a node with (dis)proof values that are inconsistent with the values of its children, the selection or expansion process is skipped and the node's values are updated. Often, this updating brings the node out of consistency with its own parents, so the process is repeated until the nodes along the current search path are locally consistent.

Constructing a Public Chemistry Benchmark

Standardized benchmarks and competitions are important tools for direct comparison of a variety of algorithmic techniques (Helmert, Do, and Refanidis 2008; Genesereth, Love, and Pell 2005). Unfortunately, no standard public benchmark exists for chemical synthesis planning. To permit principled comparison of such systems, we have created the first public chemistry benchmark. It is released to the community under a Creative Commons license and is available for download at www.cs.toronto.edu/~aheifets/ChemicalPlanning.

The benchmark contains 20 synthesis problems derived from undergraduate organic synthesis examinations. With the exception of the first problem, the benchmark problems appeared in exams at the Massachusetts Institute of Technology Course 5.13 "Organic Chemistry II" during 2001-2006. As such, the benchmark comprises problems captured from "the wild", in that the questions were not designed for computers; rather, the tasks were designed to be challenging for humans and to test fundamental chemistry concepts. The original tests and instructor-provided answer keys are distributed via MIT OpenCourseWare (MIT 2003).

We derived problem 1 from primary literature, rather than from course exams. The target of problem 1 is atorvastatin, marketed in North America as Lipitor®. It is an interesting molecule, because atorvastatin was the best-selling drug in history with yearly revenues in excess of USD$12 billion before its patent expiration in 2011. The benchmark problem uses the starting materials and reactions reported by the inventors (Brower et al. 1992; Roth 2002).

Figure 4 shows the target molecules from our benchmark, which are analogous to goal states in automated planning problems. Each synthesis problem also specifies a set of starting materials that are permitted (but not required) to be used in the construction of the target. We manually encoded these target and starting materials in a standard machine-readable format (Weininger 1988). Each problem also contains a sample solution, although multiple correct answers are generally possible. From each solution, we created a corresponding set of machine-readable reaction files. We focused on encoding the transformations enacted by the reactions and did not attempt to document reaction conditions, selectivities, or stereochemical effects. In principle, these data may be represented and future editions of the bench-

Problem	Instructor-provided solution			Proof-number search						Exhaustive search		
	operators	unique ops	depth	operators	unique ops	depth	expanded	generated	seconds	expanded	generated	seconds
1	5	5	4	5	5	4	1353	59522	10166	–	–	–
2	4	4	4	4	4	4	13	124	7	275	1897	39
3	5	5	4	5	5	4	22	145	8	176	993	26
4	4	4	4	4	4	4	25	702	15	1256	15968	224
5	3	3	2	3	3	2	11	769	31	23	1475	51
6	7	7	7	6	5	5	188	1877	31	474	4103	61
7	10	9	6	13	12	7	378	4301	70	24495	157198	2860
8	5	5	5	5	5	5	78	1810	96	474	7684	372
9	7	7	4	7	7	4	267	4614	94	2902	39256	639
10	3	3	3	3	3	3	168	3532	105	461	8985	247
11	12	12	9	12	12	9	787	15233	228	–	–	–
12	10	8	6	9	7	5	422	5932	258	9548	74865	2614
13	4	4	2	4	4	4	594	53781	3732	31	2773	121
14	4	4	3	4	4	3	406	64599	5138	149	4715	219
15	10	10	6	11	10	7	499	68335	5085	–	–	–
16	4	4	4	–	–	–	–	–	–	–	–	–
17	5	5	3	–	–	–	–	–	–	–	–	–
18	5	5	5	–	–	–	–	–	–	–	–	–
19	6	6	6	–	–	–	–	–	–	–	–	–
20	11	9	7	–	–	–	–	–	–	–	–	–

Table 1: Chemical benchmark. The instructor-provided solution describes the solutions given in the exam answer keys. Total runtime is given in seconds and includes initialization and output of solution. Dashes indicate that the problem did not complete in 6 hours.

mark could encode these reaction properties.

Table 1 describes the characteristics of the instructor-provided solutions for each problem. The 'operators' column lists the total reaction applications when the solution is flattened into a tree. The 'unique operators' column lists the number of reaction applications in the solution. When there are fewer unique operator applications than the number of operators, it means that the solution is a directed acyclic graph rather than a tree and a subproblem was reused in different parts of the synthesis. This provides an indication of the occurrence of shared synthetic intermediates that are used multiple times in the problem answer. 'Depth' describes the maximum number of reactions separating the target molecule from any starting material. An entirely linear problem (such as problems #2 and #8) will have the same number of operators, unique operators, and depth, whereas a branching solution (*e.g.*, problem #11) will have a depth that is smaller than the number of operators.

In summary, the benchmark contains 20 problems, a library of 62 unique starting materials, and 50 reaction definitions. The benchmark is designed to be stand-alone; it is expressed in standard formats and requires no external reaction or starting material libraries. The problems cover a range of difficulty, each requiring between 3 and 12 unique steps. The canonical instructor answers described solutions that include directed acyclic graphs, trees, and simple sequences. The problems were developed to be difficult for undergraduate chemistry students and, we believe, are therefore a useful testing ground for automated chemistry solvers.

Results and Discussion

Benchmark tests were performed on an IBM CL1350 cluster with 1,344 cores over 168 Infiniband-connected HS21-XM BladeServers and a DCS9550 storage system. The cluster runs the CentOS operating system, version 5.1, uses OpenJDK Runtime Environment (IcedTea6 1.7.5), and manages coarse-grained parallelism with the Portable Batch System. ChemAxon's JChem 5.7.1 was used for the molecular manipulations (ChemAxon 2011). Each test received a maximum of 6 hours CPU time and 8GB of RAM.

The benchmark provides a range of problem difficulties. Table 1 shows that 5 problems completed in less than one minute, 6 problems required between 1 minute and 5 minutes, 4 problems required between 1 hour and 3 hours, and 5 problems did not terminate within 6 hours. While problems with deep solutions generally require more time and search effort than shallow problems, the nature of the target molecule strongly impacts the search space. When many reactions can generate the substructures in a molecule, many more paths must be explored. This is illustrated in problems #2 and #4; both are linear syntheses with four steps but problem #4 generated more than 5 times as many nodes as problem #2. Furthermore, applying reactions takes a significant amount of time, which varies depending on the molecules under consideration.

Today, organic synthesis problems are solved manually; the standard commercial solver is the human. In this context, we compare against the gold standard, because the benchmark consists of problems taken from university exams intended to challenge human students. Direct comparison of our solver against other chemical planners, while desirable, is impossible. The majority of previous synthesis systems were research projects in the 1970-80s and no longer available for comparison, since even the underlying operating systems do not necessarily run on modern hardware (Corey and Wipke 1969; Agarwal, Larsen, and Gelernter 1978). The algorithms consisted of exhaustive search moderated by *ad hoc* collections of forward pruning rules.

We did compare our algorithm to a reimplementation of the algorithm used by the only existing commercial soft-

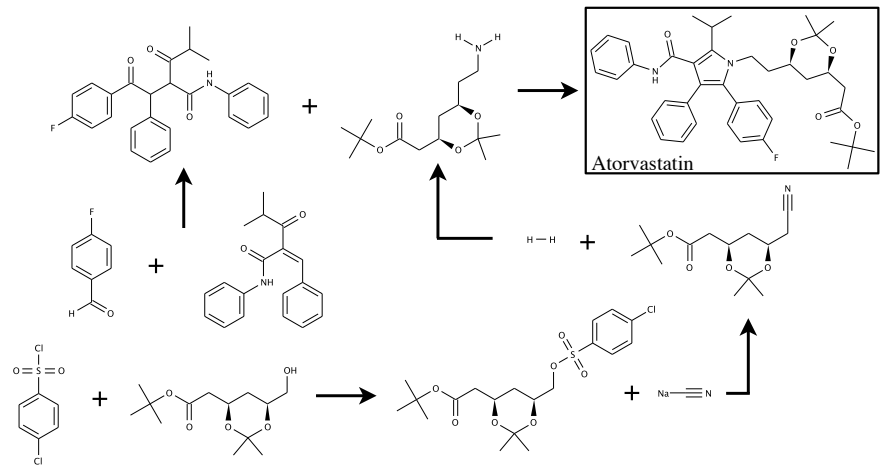

Figure 5: Computer-generated Atorvastatin synthesis matching the synthesis reported in (Brower et al. 1992) and (Roth 2002).

ware (Law et al. 2009), which is exhaustive search to a user-specified depth. We chose to reimplement the algorithm because a license costs several tens of thousands of dollars. Our implementation of IDA^* takes a user-specified bound so as to adhere as closely as possible to the published Law et al. algorithm. We specify the depth bound to be the minimum of the depths of the benchmark's provided solution and the solution returned by our proof-number searcher. Furthermore, we omitted node-construction to minimize algorithmic overhead and derive the best performance from the exhaustive algorithm.

On most problems, proof number search outperforms exhaustive search in terms of nodes expanded, nodes generated, and total time spent. For example, on problem #7, exhaustive search expanded over 64 times as many nodes as PNS. In three cases, proof-number search solved problems on which exhaustive search was terminated after 6 hours. In problems #13 and #14, PNS performed worse than exhaustive search. In these problems, the solver appears to be led away from the cheapest solution by the early discovery of a molecular precursor. As can be seen from Definitions 1 and 2, the proof number search does not optimize for shortest solutions, but rather for minimum remaining proof burden. If a particular reagent is solved early in the tree, it is considered 'proved' and the cost for using it anywhere else in the search is set to 0. Therefore, by proving early on one molecule from a complicated precursor set, the search may be driven into unfruitful areas of the search space. Integration of heuristics such as (Bertz 1981), (Hendrickson, Huang, and Toczko 1987), and (Ertl and Schuffenhauer 2009) would help avoid such waste and speed the discovery of syntheses. Such integration would widen the margin between PNS and exhaustive search as, by definition, exhaustive search can not leverage heuristics to focus the search.

We observed that our solver suffers from the problems of over- and under-counting proof and disproof numbers. These problems are observed whenever the search space is a graph rather than a tree (Seo, Iida, and Uiterwijk 2001; Kishimoto and Müller 2004) When cycles occur or when intermediate nodes are shared along different parts of the solution path, naive solutions will sum the (dis)proof numbers of leaves multiple times along overlapping paths. This causes the search to waste time in incorrect portions of the search space and may lead to infinite looping (Kishimoto 2005; Kishimoto and Müller 2008). Solutions that avoid both high computational overhead and inaccurate proof and disproof counts are an area of active research (Kishimoto and Müller 2003; Kishimoto 2005; 2010).

Conclusion and Future Work

This paper introduces an economically- and medically-critical problem to the AI community. It is clear that our solver is not yet competitive with a skilled human undergraduate chemistry student. However, it can already solve a number of challenging synthesis problems, including the the expert-derived synthesis of atorvastatin, the best selling drug in history.

A move from theory and prototype to a realistic system requires several extensions. Search efficiency could be improved by initializing the AND/OR (dis)proof numbers with a stronger heuristic. The reaction definitions should be expanded to encompass a broader set of reactions and the starting materials library should be scaled to a full commercial chemical database. These additional reactions and starting materials would permit the discovery of novel syntheses without blocking the recognition of the synthesis we report in Figure 5.

Despite these limitations, our prototype shows the feasibility of applying planning techniques to the chemical synthesis problem. We described the first application of proof-number search to chemical synthesis planning and the first benchmark for comparing synthesis solvers. We hope chemical synthesis planning will be a fruitful domain for AI research. To that end, we are distributing our code and benchmark, which are available at www.cs.toronto.edu/~aheifets/ChemicalPlanning.

Acknowledgements

We thank Kristen Fortney, Max Kotlyar, and Izhar Wallach for their valuable comments on earlier drafts. Computational

analysis was supported in part by Canada Foundation for Innovation [CFI #12301 and CFI #203383] and Ontario Research Fund [GL2-01-030]. IJ is supported in part by the Canada Research Chair Program. This research was funded in part by the Ontario Ministry of Health and Long Term Care. The views expressed do not necessarily reflect those of the OMOHLTC.

References

Agarwal, K.; Larsen, T.; and Gelernter, H. 1978. Application of chemical transforms in SYNCHEM2, a computer program for organic synthesis route discovery. *Comp. & Chemistry* 2(2):75–84.

Allis, L.; van der Meulen, M.; and van den Herik, H. 1994. Proof-number search. *Artificial Intelligence* 66:91–124.

Bertz, S. H. 1981. The first general index of molecular complexity. *J. Am. Chem. Soc.* 103:3599–3601.

Bonet, B., and Geffner, H. 2006. Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. In *ICAPS*, 142–151. AAAI.

Brower, P. L.; Butler, D. E.; Deering, C. F.; Le, T. V.; Millar, A.; Nanninga, T. N.; and Roth, B. D. 1992. The synthesis of (4R-cis)-1,1-dimethylethyl 6-cyanomethyl-2,2-dimethyl-1,3-dioxane-4-acetate, a key intermediate for the preparation of CI-981, a highly potent, tissue selective inhibitor of HMG-CoA reductase. *Tetrahedron Letters* 33(17):2279–2282.

Cao, Y.; Jiang, T.; and Girke, T. 2008. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. *Bioinformatics* 24(13):i366–74.

Carey, J. S.; Laffan, D.; Thomson, C.; and Williams, M. T. 2006. Analysis of the reactions used for the preparation of drug candidate molecules. *Org. Biomol. Chem.* 4:2337–2347.

ChemAxon. 2011. JChem 5.7.1. *http://www.chemaxon.com.*

Clayden, J.; Greeves, N.; Warren, S.; and Wothers, P. 2000. *Organic Chemistry*. Oxford University Press.

Corey, E. J., and Wipke, W. T. 1969. Computer-Assisted Design of Complex Organic Syntheses. *Science* 166(3902):178–192.

Daylight. 2008. Daylight Chemical Information Systems, Inc. *http://www.daylight.com/daycgi_tutorials/smirks_examples.cgi*.

Ertl, P., and Schuffenhauer, A. 2009. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment comtributions. *Cheminformatics* 1(8).

Genesereth, M. R.; Love, N.; and Pell, B. 2005. General game playing: Overview of the AAAI competition. *AI Magazine* 26(2):62–72.

Helmert, M.; Do, M.; and Refanidis, I. 2008. International planning competition. *http://ipc.informatik.uni-freiburg.de/Results*.

Hendrickson, J. B.; Huang, P.; and Toczko, A. G. 1987. Molecular complexity: a simplified formula adapted to individual atoms. *Journal of Chemical Information and Computer Sciences* 27(2):63–67.

Homem de Mello, L., and Sanderson, A. 1990. AND/OR graph representation of assembly plans. *IEEE Transactions on Robotics and Automation* 6(2):188 –199.

Hudlicky, T., and Reed, J. W. 2007. *The Way of Synthesis: Evolution of Design and Methods for Natural Products*. Wiley-VCH.

Kishimoto, A., and Müller, M. 2003. Df-pn in go: An application to the one-eye problem. In van den Herik, H. J.; Iida, H.; and Heinz, E. A., eds., *ACG*, volume 263 of *IFIP*, 125–142. Kluwer.

Kishimoto, A., and Müller, M. 2004. A general solution to the graph history interaction problem. *AAAI* 644–649.

Kishimoto, A., and Müller, M. 2008. About the completeness of depth-first proof-number search. In van den Herik, H. J.; Xu, X.; Ma, Z.; and Winands, M. H. M., eds., *Computers and Games*, volume 5131 of *Lecture Notes in Computer Science*, 146–156. Springer.

Kishimoto, A. 2005. *Correct and Efficient Search Algorithms in the Presence of Repetitions*. Ph.D. Dissertation, University of Alberta.

Kishimoto, A. 2010. Dealing with infinite loops, underestimation, and overestimation of depth-first proof-number search. *AAAI*.

Landrum, G. 2006. Rdkit: Open-source cheminformatics. *http://www.rdkit.org*.

Law, J.; Zsoldos, Z.; Simon, A.; Reid, D.; Liu, Y.; Khew, S. Y.; Johnson, A. P.; Major, S.; Wade, R. A.; and Ando, H. Y. 2009. Route designer: A retrosynthetic analysis tool utilizing automated retrosynthetic rule generation. *J. Chem. Inf. Model* 49(3):593–602.

Liang, Q. A., and Su, S. Y. W. 2005. AND/OR graph and search algorithm for discovering composite web services. *Int. J. Web Service Res.* 2(4):48–67.

MIT. 2003. MIT OpenCourseWare. *http://ocw.mit.edu*.

Nagai, A. 1998. A new AND/OR Tree Search Algorithm Using Proof Number and Disproof Number. In Frank, I.; Matsubara, H.; Tajima, M.; Yoshikawa, A.; Grimbergen, R.; and Müller, M., eds., *Complex Games Lab Workshop*, 40–45. Electrotechnical Laboratory, Machine Inference Group, Tsukuba, Japan.

Nagai, A. 2002. *Df-pn Algorithm for Searching AND/OR Trees and Its Applications*. Ph.D. Dissertation, University of Tokyo.

Nicolaou, K. C., and Sorensen, E. J. 1996. *Classics in Total Synthesis: Targets, Strategies, Methods*. Wiley-VCH.

O'Boyle, N.; Banck, M.; James, C.; Morley, C.; Vandermeersch, T.; and Hutchison, G. 2011. Open babel: An open chemical toolbox. *Journal of Cheminformatics* 3(1):33.

Pirok, G.; Máté, N.; Varga, J.; Szegezdi, J.; Vargyas, M.; Dóránt, S.; and Csizmadia, F. 2006. Making "real" molecules in virtual space. *Journal of Chemical Information and Modeling* 46(2):563–568. PMID: 16562984.

Roth, B. D. 2002. The discovery and development of atorvastatin, a potent novel hypolipidemic agent. In King, F.; Oxford, A.; Reitz, A. B.; and Dax, S. L., eds., *Progress in Medicinal Chemistry*, volume 40. Elsevier. 1 – 22.

S. G. Gregory et al. 2006. The DNA sequence and biological annotation of human chromosome 1. *Nature* 441(7091):315–321.

Schaeffer, J.; Burch, N.; Bjornsson, Y.; Kishimoto, A.; Muller, M.; Lake, R.; Lu, P.; and Sutphen, S. 2007. Checkers is solved. *Science* 317(5844):1518–1522.

Seo, M.; Iida, H.; and Uiterwijk, J. W. H. M. 2001. The PN*-search algorithm: Application to tsume-shogi. *Artif. Intell.* 129(1-2):253–277.

Todd, M. H. 2005. Computer-aided organic synthesis. *Chem. Soc. Rev.* 34:247–266.

Wang, Y.; Xiao, J.; Suzek, T. O.; Zhang, J.; Wang, J.; and Bryant, S. H. 2009. PubChem: a public information system for analyzing bioactivities of small molecules. *Nucleic Acids Research* 37(suppl 2):W623–W633.

Weininger, D. 1988. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences* 28(1):31–36.